(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété Intellectuelle
Bureau international

(43) Date de la publication internationale
16 octobre 2003 (16.10.2003)

PCT

(10) Numéro de publication internationale
WO 03/085521 A1

(51) Classification internationale des brevets[7] : G06F 9/44

(21) Numéro de la demande internationale :
PCT/FR03/01019

(22) Date de dépôt international : 2 avril 2003 (02.04.2003)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
02/04235 5 avril 2002 (05.04.2002) FR

(71) Déposant et
(72) Inventeur : **AUBERLET de CHELLE, Yvonne** [FR/FR]; 2, allée de Bernadotte, F-92330 Sceaux (FR).

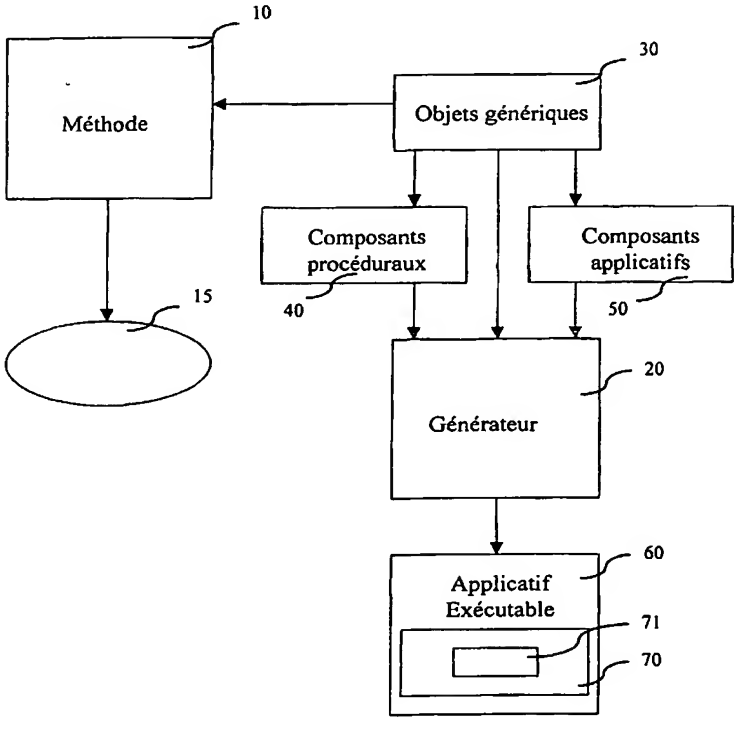
(72) Inventeurs; et
(75) Inventeurs/Déposants *(pour US seulement)* : **DELLE-VEDOVE, Agnès** [FR/FR]; 8, avenue René Isidore, F-92260 Fontenay Aux Roses (FR). **DELLE-VEDOVE, Serge** [FR/FR]; 8, avenue René Isidore, F-92260 Fontenay Aux Roses (FR).

(74) Mandataire : **RAVINA, Bernard**; Ravina S.A., 8, rue des Briquetiers, B.P. 77, F-31700 Blagnac Cedex (FR).

[Suite sur la page suivante]

(54) Title: METHOD AND DEVICE FOR GENERATING SOFTWARE WITH CUSTOMIZED EXECUTION AND UPGRADABLE WITHOUT COMPUTER PROGRAMMING

(54) Titre : PROCEDE ET DISPOSITIF DE GENERATION DE LOGICIELS EXECUTABLES SUR MESURE ET EVOLUTIFS SANS PROGRAMMATION INFORMATIQUE



10...METHOD
30...GENERIC OBJECTS
40...PROCEDURAL COMPONENTS
50...APPLICATIVE COMPONENTS
20...GENERATOR
60...EXECUTABLE APPLICATIVE SOFTWARE

(57) Abstract: The invention concerns a method for generating applicative software for management of a process using a system software common to all the applicative software. It comprises: a step of representing the process (32), using a very small number of classes of actions or generic objects, typically less than twenty, in at least a diagram of the application; a step of transcribing (33) each object of each diagram into an action corresponding to an object provided with attributes, each class of action or generic object being, during the transcription step, associated with an application data input interface; a step of transcribing the nodes, branches and end-nodes (34) of each diagram into an action corresponding to an object provided with attributes, a step of pre-compiling (35) which consists in verifying whether the attributes of the objects required for the application operational logic exist and are properly supplied, in syntax; a compilation (36) which consists in integrating and assembling the data specifications of the objects provided with attributes with the system software, to obtain an executable applicative software; and a step of executing (37) the executable applicative software.

[Suite sur la page suivante]

WO 03/085521 A1

(81) **États désignés** *(national)* : AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) **États désignés** *(régional)* : brevet ARIPO (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), brevet eurasien (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), brevet européen (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), brevet OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Publiée :

— *avec rapport de recherche internationale*

— *avant l'expiration du délai prévu pour la modification des revendications, sera republiée si des modifications sont reçues*

En ce qui concerne les codes à deux lettres et autres abréviations, se référer aux "Notes explicatives relatives aux codes et abréviations" figurant au début de chaque numéro ordinaire de la Gazette du PCT.

(57) Abrégé : Le procédé de génération de logiciel applicatif de gestion d'un processus met en oeuvre un logiciel système commun à tous les logiciels applicatifs. Il comporte: une étape de représentation du processus (32), mettant en oeuvre un très petit nombre de classes d'actions ou objets génériques, typiquement inférieur à vingt, dans au moins un diagramme de l'application, une étape de transcription (33) de chaque objet de chaque diagramme en une action correspondant à un objet doté d'attributs, chaque classe d'action ou objet générique étant, pendant l'étape de transcription, associé à une interface de saisie de données d'application, une étape de transcription des noeuds, embranchements et feuilles (34) de chaque diagramme en une action correspondant à un objet doté d'attributs, une étape de pré-compilation (35) au cours de laquelle on vérifie que les attributs des objets nécessaires pour la logique de fonctionnement de l'application existent et sont convenablement fournis, en syntaxe, une étape de compilation (36) au cours de laquelle les descriptifs de données des objets dotés d'attributs sont intégrés et sont assemblés avec le logiciel système, pour obtenir un logiciel applicatif exécutable, et une étape d'exécution (37) du logiciel applicatif exécutable.

PROCEDE ET DISPOSITIF DE GENERATION DE LOGICIELS EXECUTABLES SUR MESURE ET EVOLUTIFS SANS PROGRAMMATION INFORMATIQUE

La présente invention vise un procédé et dispositif de génération de logiciels exécutables sur mesure et évolutifs sans programmation informatique. Elle s'applique en particulier à la génération de logiciels de gestion et de pilotage d'un centre de profits.

Le processus de génération de logiciels actuellement connu est basé sur la négociation d'un cahier des charges entre le client et l'informaticien chargé de réaliser une application informatique répondant au cahier des charges, puis la programmation par un informaticien d'un logiciel correspondant au cahier des charges négocié. Le client ne maîtrisant pas les contraintes techniques qui encadrent le travail de l'informaticien, la négociation est asymétrique et provoque, par elle-même, une insatisfaction du client. En outre, toute modification ultérieure du logiciel est soumise à un nouveau travail de l'informaticien, ce qui gêne l'évolution de ce logiciel.

La présente invention vise à remédier à ces inconvénients. En particulier, la présente invention vise à éviter l'intervention d'un informaticien dans la programmation ou la modification d'un logiciel. En d'autres mots, la présente invention vise à donner au client :

- la maîtrise du processus opérationnels de génération de l'application,
- d'éviter que l'application soit soumise aux modèles sommaires donnés par les informaticiens,
- d'obtenir, en quelques minutes, une application informatique sur mesure, répondant intégralement à ses voeux, sans avoir à maîtriser de langage informatique.

Selon un aspect, la présente invention vise un procédé de génération de logiciel applicatif de gestion d'un processus, caractérisé en ce qu'il met en oeuvre un logiciel système commun à tous les logiciels applicatifs et en ce qu'il comporte :

- une étape de représentation du processus, mettant en oeuvre un très petit nombre de classes d'actions ou objets génériques, typiquement inférieur à vingt, dans au moins un diagramme de l'application,

- une étape de transcription de chaque objet de chaque diagramme en une action correspondant à un objet doté d'attributs, chaque classe d'action ou objet générique étant, pendant l'étape de transcription, associé à une interface de saisie de données d'application,
- une étape de transcription des noeuds, embranchements et feuilles de chaque diagramme en une action correspondant à un objet doté d'attributs,
- une étape de pré-compilation au cours de laquelle on vérifie que les attributs des objets nécessaires pour la logique de fonctionnement de l'application existent et sont convenablement fournis, en syntaxe,
- une étape de compilation au cours de laquelle les descriptifs de données des objets dotés d'attributs sont intégrés et sont assemblés avec le logiciel système, pour obtenir un logiciel applicatif exécutable, et
- une étape d'exécution du logiciel applicatif exécutable.

Grâce à ces dispositions, le client effectue la représentation du processus correspondant à l'application par une simple représentation graphique en diagramme, qui définit intégralement l'application informatique qu'il souhaite, sans autre contrainte que celle de connaître le très petit nombre d'objet génériques à employer. Dès que cette phase est achevée, la transcription peut être effectuée par simple saisie, par une personne de faible qualification. Les deux étapes restantes, pré-compilation et compilation sont totalement automatiques.

On observe que les descriptifs d'applications peuvent être modifiés ou complétés aisément : une nouvelle compilation fournit un exécutable à jour, compatible avec les autres versions, et évolutif, en quelques minutes.

On observe qu'assembler un certain nombre d'actions standards en un diagramme donne un résultat plus simple à contrôler que du code (par exemple du code "C") généré directement. Le principe est que des actions, qui travaillent sur des données convenablement organisées, peuvent être considérées comme sûres, et leur contenu jamais tracé, ce qui évite de longues phases de debuggage.

Les déroulements de ces diagrammes sont fixés de façon permanente dans des procédures standard. Il suffit de fournir les identificateurs d'écrans et de fichiers utilisés, et ces diagrammes vont les visualiser, modifier les menus au fur et à mesure des étapes, etc ... Le mécanisme de la procédure est toujours identique, mais son contenu est adapté, sur mesure, aux besoins exprimés par les utilisateurs. Bien

entendu, un certain nombre de points d'entrée subsiste pour insérer des actions particulières à chaque client, par exemple si le mode de lecture des données est très particulier (par exemple, si le fichier n'est pas un fichier standard mais est importé d'un autre système, la façon de lire la première fiche n'est pas standard, donc doit être donnée par le concepteur). Par contre, le fait d'avoir le choix de demander la première fiche reste standard, donc le diagramme considéré est utilisable. L'ensemble des informations propres à chacun des applicatifs est réuni de façon structurée dans des "cartes" : le concepteur "câble" une carte pour approprier le processus standard à chaque processus particulier.

Ce principe permet, par exemple, à une même action de visualisation d'afficher des écrans différents suivant la carte dans laquelle on se trouve, tout en conservant les mêmes paramètres d'appel. Il suffit d'avoir modifié auparavant une table de paramètres indexés.

Ce type d'interprétation est fait non pas par l'action elle-même, mais par le mécanisme d'interprétation des diagrammes ("câblage" de la carte). En d'autres termes, une action comporte des données soit décrites de façon spécifique soit décrites par des paramètres (carte + numéro dans la carte).

Selon des caractéristiques particulières, au cours de l'étape de transcription d'objets, au moins une action lance un traitement complet se trouvant à un endroit distant d'une arborescence correspondant audit au moins un diagramme, et une fois celui-ci terminé, retourne à son point de départ.

Grâce à ces dispositions, les diagrammes peuvent être constitués, modifiés et mis à jour indépendamment.

Selon des caractéristiques particulières, au cours de l'étape d'exécution, le logiciel applicatif exécutable met en oeuvre une bibliothèque de gestion du déroulement des processus correspondant audit au moins un diagramme, ladite bibliothèque constituant un automate qui gère le déroulement des processus et exécute les opérations qui les jalonnent, le déroulement des opérations étant défini, dans le référentiel applicatif, à l'aide de la méthode, en décrivant les flux réels.

Ainsi, la seule mise en oeuvre de la méthode et de la transcription permet de programmer une application.

Selon des caractéristiques particulières, au cours de l'étape de compilation ou au cours de l'étape d'exécution, il met en oeuvre un moteur qui comporte un superviseur chargé de reconnaître la configuration matérielle et de communication.

Grâce à ces dispositions, il est inutile de générer la configuration dans l'applicatif. La gestion des échanges se fait par des tâches sur l'ensemble d'un écran, sur certains champs ou par des messages envoyés à l'utilisateur par le déroulement du processus.

Selon des caractéristiques particulières, le moteur gère une ou plusieurs bases de données d'après un descriptif des fichiers de données fourni par le référentiel applicatif, c'est-à-dire une liste des informations contenues dans chaque fiche et la liste des index d'accès, avec une liste des champs servant à constituer chacun de ces index, les liens entre des codifications multiples d'un même item dans plusieurs services, plusieurs sites ou plusieurs entreprises.

Selon des caractéristiques particulières, les bases de données sont synchronisées selon une fréquence déterminée par le diagramme, à la demande ou avant certains événements prédéterminés.

Ainsi, les bases de données peuvent être synchronisées toutes les nuits, lorsqu'un utilisateur autorisé le demande ou pour un événement le nécessitant, par exemple pour réaliser un devis client (les stocks et les prix instantanés devant, par exemple, être connus pour cette opérations).

Selon des caractéristiques particulières, l'étape de transcription d'objets comporte, pour programmer chaque action :

- une étape de nommage au cours de laquelle on donne un nom à ladite action,
- une étape de définition de fonction, au cours de laquelle on met ladite action en correspondance avec une tâche, et,
- une étape de définition d'information, au cours de laquelle on désigne les informations qui seront traitées dans l'action.

Les actions sont enchaînées suivant leur ordre fonctionnel dans des diagrammes. Les actions ramenant toutes à des codes définis, il est possible de représenter ainsi les structures de contrôle classiques telles que branchement, boucle, test, etc ... Ces contrôles sont eux-mêmes effectués par des actions.

Selon des caractéristiques particulières, l'étape de compilation remplace le nom d'action donné, au cours de l'étape de nommage, par le transcripteur, par un index dans une table de tâches.

Grâce à ces dispositions, l'étape de transcription peut être assez intuitive, le transcripteur choisissant un nom d'action qui lui semble aisément reconnaissable, tout en permettant un fonctionnement automatique et efficace de l'application compilée.

D'autres avantages, buts et caractéristiques de la présente invention ressortiront de la description qui va suivre, faite en regard des dessins annexés dans lesquels :

- la figure 1 représente, schématiquement, les composants du système objet de la présente invention,

- la figure 2 représente, schématiquement, des composants mis en oeuvre par le procédé objet de la présente invention, et

- la figure 3 représente, schématiquement, les étapes mises en oeuvre par le procédé objet de la présente invention.

Dans la figure 1 sont représentés une méthode 10 de description de processus et de génération de diagramme 150, un générateur d'application informatique 20 mettant en oeuvre des objets génériques 30, des composants procéduraux 40 et des composants applicatifs 50 pour générer un logiciel exécutable 60 comportant un automate 70 mettant en oeuvre les composants applicatifs 50.

Conformément à la méthode 10, méthode de description cybernétique des processus, de quelque nature qu'ils soient, pour obtenir un descriptif sous forme de macro-langage, appelé "référentiel applicatif", le client définit d'abord sont processus en grands ensembles, en décrivant leurs relations (par exemple gestion des stocks, comptabilité, décision d'organisation de travail, prise de commande, alertes nécessaires ...). Dans un second temps, chaque ensemble est considéré et décomposé en sous-ensembles de tâches, et ainsi que suite jusqu'à ce que toutes les actions du processus soient décrites en ne mettant en oeuvre que des diagrammes composés d'actions correspondant, chacune à un d'objet générique de l'une des classes suivantes, doté d'attributs nécessaires au fonctionnement des diagrammes :

a) entreposage structuré de données,

b) transmission de données,

c) supervision,

d) communication avec les utilisateurs,

e) déroulement du processus,

f) système opérationnel (capteurs de données et de signaux, émission de messages et d'alertes),

g) pilotage à deux niveaux (prévisionnel ou opérationnel).

Le macro-langage défini par la méthode 10 décrit les processus pour constituer une couche logique, au dessus de la couche des programmes informatiques. A la base de la création de ce macro-langage, il a été créé une typologie des traitements informatiques.

On observe que le diagramme lui-même est composé d'une racine, d'embranchements, de noeuds et de feuilles, chacun de ces composants étant aussi représenté par une action.

Avec cette méthode 10, le client définit les limites du problème/processus pour lequel il souhaite mettre en oeuvre un logiciel exécutable sur mesure et ses relations avec l'environnement. Il peut y introduire exactement ce que font les différentes personnes du centre de profit, les événements exceptionnels prévus (par exemple les vacances de collaborateurs, les arrêts d'une machine en maintenance), les aléas (par exemple une commande urgente ou une panne machine) dont il souhaite tenir compte et les critères de décision d'alerte dont ils souhaitent disposer.

Pour déterminer la typologie indiquée ci-dessus, les opérations répétitives réalisées au cours du déroulement d'une application informatique ont été recensées :

- ces opérations sont identiques, quel que soit le sujet traité,
- elles sont en nombre limité,
- elles peuvent être réparties en sept classes (classes "a" à "g", ci-dessus)

Chacune des sept classes d'opérations est traitée selon une approche orientée « objet », en encapsulant les règles internes propres à la classe, règles de nature déterminée répétitive, s'agissant du comportement d'objets informatiques (par opposition aux objets des domaines applicatifs de gestion, soumis à l'aléatoire). Les caractéristiques propres à chaque classe, et nécessaires pour faire fonctionner les règles internes, ont été inventoriées et réunies dans des « structures » d'attributs, au sens informatique du terme « structure ».

Ainsi, dans une deuxième étape, les diagrammes représentant le déroulement du processus de gestion ou de pilotage sont saisis en mettant en oeuvre des écrans de saisie qui guident l'opérateur et lui interdisent de faire certaines saisies incorrectes ou incomplètes. Chaque objet est doté d'attributs définissant des données dans des formats prédéterminés (types de données, quantité, longueur et unité), des éléments de stockage des données (dans des fichiers permanents et/ou avec une date de validité), des droits d'accès des différents utilisateurs et les interfaces qu'ils utiliseront, des éléments de décision avec comparaison d'options (décision : acheter ou fabriquer, par exemple). Un support numérique permet de représenter une application en remplissant les champs d'attributs dans les structures liées à chaque classe d'objet (par exemple, un écran est totalement défini dans deux structures qui décrivent l'ensemble de l'écran et chaque champ de l'écran).

Le générateur 20 effectue les opérations suivantes :

- il vérifie la syntaxe,
- il vérifie l'existence des informations nécessaires,
- il génère la base de données pour l'application, et
- il compile l'exécutable pour fournir une application informatique sur mesure

Le générateur 20 effectue une pré-compilation qui vérifie que les attributs nécessaires pour la logique de fonctionnement de l'application existent et sont convenablement remplis (en syntaxe et non en contenu).

Chaque action est associée à une interface de saisie. Cette particularité est un des aspects de la présente invention. La compilation accomplit ensuite l'intégration des descriptifs de données d'application et les assemble avec le logiciel système, pour obtenir un exécutable. Le logiciel système est permanent, commun à toutes les applications.

L'automate 70 comporte un moteur 71 et met en oeuvre des fonctionnalités. Le moteur 71 représente l'ordre de mise en oeuvre des fonctionnalités et les procédures correspondant aux diagrammes de déroulement de l'application informatique. Le moteur 71 relie chaque action à l'une de ses tâches types 30 (il y en a une soixantaine) au moyen de mécanismes d'exécution répartis en sept classes.

Le moteur 71 possède un ensemble prédéterminé et permanent de tâches programmées informatiquement, performantes et contrôlées, appelées dans des actions applicatives. Ces actions sont exécutées par le coeur du moteur 71, suivant

un ordre défini dans des diagrammes organisés sous forme d'arborescence, avec un mécanisme de contrôle du retour des tâches conduisant à la poursuite du diagramme en cours, ou à un branchement vers un autre diagramme ou un autre sous processus. Par exemple, une tâche de calcul se poursuit sur la tâche suivante dans le diagramme en cours, alors qu'une tâche de comparaison conduit généralement à un branchement défini par la description de l'action appelant la tâche de comparaisons.

On observe que les descriptifs d'applications peuvent être modifiés ou complétés : une nouvelle compilation fournit un exécutable à jour, compatible avec les autres versions, et évolutif. Cependant, si on change le descriptif d'un format de données, il faut transférer les données dans un nouveau fichier, avec le nouveau format, ceci grâce à une procédure automatique appartenant aux composants procéduraux 40. En revanche, on peut ajouter des champs, dans la limite de la place disponible dans le fichier, ou étendre les dimensions du fichier avec la procédure automatique évoquée ci-dessus.

Grâce à la mise en oeuvre de la présente invention, on transforme une description procédurale en exécutable sans la moindre ligne de langage informatique. Selon un de ses aspects, la présente invention définit un macro-langage et/ou une nouvelle couche système.

On comprends aisément que, lorsque le client souhaite une modification, on modifie le schéma du diagramme et, après traitement, il obtient un nouvel exécutable sans qu'il soit nécessaire de mettre en oeuvre des compétences d'informaticien.

Dans les logiciels connus, le déroulement des échanges entre le logiciel et les utilisateurs est conduit par le système graphique. Ici, c'est le logiciel qui a la maîtrise du déroulement des opérations, y compris avec le système graphique. On obtient ainsi l'indépendance par rapport aux systèmes externes, la possibilité de communiquer indifféremment avec différentes interfaces, par identification de l'interlocuteur.

Le moteur 71 exécute les opérations du processus d'une façon neutre et indépendante du domaine d'application, grâce à des mécanismes d'exécution issus d'une typologie rigoureuse des modes de fonctionnement de l'informatique et répartis en sept classes.

On observe que, dans la douzaine de structures utilisées (structure au sens informatique du terme), trois sont des structures dynamiques, une est une structure de navigation et les autres sont des structures descriptives de données.

Des objets procéduraux applicatifs répétitifs et communs à tous types d'application sont utilisés : saisie de fiche simple, saisie de fiche avec liste attachée (par exemple les mouvements de stock pour un produit), impression, alerte, reconfiguration de fichiers de données, liaisons avec d'autres systèmes d'information. S'y rattachent un mécanisme de déroulement et un superviseur qui constituent le "coeur" du moteur 71.

On va maintenant décrire d'autres objets mis en oeuvre conformément à la présente invention.

1/ Les données (ou "rubriques")

- nom de la donnée
- type alphanumérique, nombre
- taille
- cadrage
- nombre de décimales et signe (si donnée numérique)

On observe qu'une rubrique est définie une seule fois pour une application et peut être appelée dans tout fichier ou interface de cette application. Une rubrique porte le même nom dans toutes les applications, mais sa description est adaptée aux besoins de chaque applicatif : automatiquement les fichiers, interfaces et traitements disposent des spécifications propres à l'application.

2/ Les fichiers de données (ou tables)

- le nombre de données dans le fichier et la dimension de l'espace sur disque ou en mémoire pour la gestion de chaque fiche,
- la description des données de chaque enregistrement du fichier et leurs liens avec des données d'autres fichiers ou écrans
    - o nom de la rubrique,
    - o liens avec d'autres fichiers ou écrans

3/ Les interfaces utilisateurs (écrans)

- le nombre de données dans l'écran et la dimension de l'espace de gestion de l'écran

- la description de chaque donnée de l'écran fichier, de son comportement dans les échanges à travers l'interface homme-machine et ses liens avec des données d'autres fichiers ou écrans :
    - o nom de la rubrique,
    - o liens avec d'autres fichiers ou écrans,
    - o type d'objet interface homme machine : libellé, texte, boutons radio, ...
    - o positionnement sur l'écran,
    - o mode de saisie : en mode création de clé (mode d'accès au fichier, par exemple par nom ou par localisation) ou de sélection, en mode courant (dans lequel on ne modifie plus de clé, sauf si elle est modifiable),
    - o traitements liés à la donnée, en début et en fin de saisie (à fin de contrôle, de traitements, ...),
    - o champ sur lequel retourner en cas d'erreur

4/ Les espaces de gestion dynamique des fichiers et écrans, lors de leur utilisation

- dimension des espaces,
- valeur des données de la fiche ou de l'écran en cours d'usage (identification des données, quantités, montants, ...),
- états des traitements (phases de recherche d'une fiche, de traitements de la fiche, de validation de fiche),

5/ Un navigateur, traçant pour chaque utilisateur, la route suivie au cours du déroulement de son travail et assistant une hyper-navigation par la possibilité d'atteindre directement des processus distants, puis de revenir au processus en cours :

- étapes du parcours,
- travaux réalisés au cours de l'étape, avec un traçage des liens avec les processus distants, assorti d'un mécanisme de retour automatique en arrière,

6/ Des diagrammes de flux

- description de diagrammes de traitements, avec des branchements soit choisis par l'utilisateur (par exemple par utilisation de menus), soit imposés par la logique du contexte (aiguillage sur deux processus différents de calcul d'un prix de revient suivant le caractère acheté ou fabriqué d'un produit),

- sur chaque branche, appel à des actions de traitement : calcul, affectation de valeurs à des données, comparaisons, interface avec des fichiers ou des écrans, ...

7/ Des messages et menus

- messages d'information,
- messages d'alerte,
- choix de traitements par des menus

D'une manière générale, la présente invention vise un système logiciel composite associant :

1/ un moteur 71 capable de réaliser de façon standard et en temps réel :

- des procédures de traitement d'informations de toutes natures, soit en simulation d'hypothèses futures, soit en mettant en œuvre les événements réels,
- sous forme d'ensembles et sous-ensembles de traitements, coordonnés et possédant des capacités de boucles de feed-back fournissant et/ou échangeant des informations en retour,
- avec la mise à disposition de tableaux de bord permanents et le déclenchement de signaux d'alerte présentés instantanément ou en temps utile aux utilisateurs concernés (on définit donc qui doit être alerté et quand, dans le diagramme).

2/ un référentiel applicatif, créé à l'aide de la méthode 10, décrivant les informations et leurs traitements spécifiques et répondant exactement aux besoins de l'entreprise et des utilisateurs :

- pour un processus de gestion, ou un ensemble de processus en interrelation,
- sous une forme de système et sous-systèmes avec des boucles de feed-back,
- avec une description sous forme banale et non informatique des flux réels, si nécessaire aménagés pour plus de performances et de réactivité des acteurs et du système dans son ensemble.

3/ un générateur 20 de logiciel exécutable sur mesure :

- le logiciel exécutable est généré sans programmation, en associant le moteur 71 (paragraphe 1) et le référentiel (paragraphe 2) de l'applicatif,
- la description sur mesure des besoins exprimés dans le référentiel est traduite automatiquement dans le langage du moteur 71

- le générateur de logiciel exécutable est lui-même construit avec le moteur 71 standard avec son propre référentiel applicatif.

D'une manière plus détaillée, le moteur 71 est un ensemble de bibliothèques de fonctions C, java et Internet, permettant une gestion de processus, simples ou complexes, avec :

- un traitement en temps réel des événements et des informations qui leur sont liées,
- la détection de situations anormales accompagnées d'un système de signaux d'alerte,
- l'identification complète des événements avec un stockage des données sous forme d'une base de connaissances à jour en permanence.

Chaque bibliothèque est spécialisée dans un domaine particulier, et liée aux autres dans un schéma de relations fonctionnelles. Les fonctionnalités et données utilisées dans chaque bibliothèque sont formalisées dans un langage et avec une syntaxe propre au moteur 71.

La bibliothèque de gestion du déroulement des processus constitue l'automate 70 qui gère le déroulement des processus et exécute les opérations qui les jalonnent. Le déroulement des opérations est défini dans le référentiel applicatif, à l'aide de la méthode 10, en décrivant les flux réels, si nécessaire aménagés pour plus de performances et de réactivité, dès la conception et facilement adaptables lorsque l'organisation change dans le temps.

Les processus sont structurés sous forme d'arborescences de flux opérationnels. Ces arborescences représentent les flux du réel, tels qu'ils ont été analysés avec les utilisateurs à l'aide de la méthode 10. Le moteur 71 exécute l'enchaînement des opérations sur chaque branche du flux en cours de traitement.

Des branchements sont déclenchés :

- soit par un choix de l'utilisateur à l'aide d'un menu,
- soit par évaluation de la situation conditionnant les étapes suivantes :
  - o choix d'une branche parmi plusieurs pour poursuivre le processus,
  - o mise en œuvre de signaux d'alerte et de boucles de feed-back (asservissement)

Chaque processus est jalonné d'opérations ou actions. Le moteur 71 exécute la suite de ces actions en faisant appel à une bibliothèque de tâches.

Les tâches utilisées, illustrées en figure 2, associées au déroulement des flux de la réalité, sont de cinq types :

a) tâches complexes 21 d'appel à d'autres processus ou sous-processus,
b) tâches complexes d'aiguillage 22,
c) tâches simples 23 de calculs, de traitements d'informations,
d) tâches de communication et d'échanges 24 avec les utilisateurs,
e) tâches de transactions 25 avec des bases de données.

Chacun de ces cinq types va être décrit ci-dessous.

a) Tâches complexes 211 d'appel à d'autres processus ou sous-processus

- ces tâches lancent des applications, sous forme d'un nom de processus, chaque processus étant assorti de droits d'accès ;
- chaque personne ayant un code d'accès reçoit
  - o l'attribution d'un processus initial par lequel elle entre dans le système,
  - o une définition de ses droits qui guident ses accès à d'autres processus en cours de navigation,
- les tâches d'appel à des processus permettent soit d'accéder à des « briques » fonctionnelles communes, soit de pratiquer une « hyper navigation » entre des fonctions et processus différents.
  - o Le branchement sur un autre processus se termine par le retour automatique sur le processus en cours (hyper-navigation)
- les « tâches d'appel de processus » permettent de rendre très simple et souple l'utilisation commune de processus et de déclencher quatre types de processus :
  - o un sous-processus 211 faisant suite au processus en cours,
  - o un processus distant 212,
  - o un sous-processus standard 213, commun à plusieurs processus,
  - o un applicatif générique complexe 214, commun à toutes les transactions avec les utilisateurs

Un sous-processus 211 se déclenche à la suite d'un aiguillage issu d'un choix de l'utilisateur par utilisation d'un menu (le sous-processus a le nom du processus origine, complété du numéro du choix dans le menu) ou d'une évaluation de la situation (le référentiel applicatif peut soit donner un nom spécifique au sous-

processus, soit conserver le nom du processus origine, en le complétant d'une lettre (par exemple un « e » en cas de détection et de traitement d'une erreur).

Par un processus distant 212, l'utilisateur évite un parcours difficile à travers des suites de menus. L'utilisateur accède directement à des tâches annexes par une « hyper navigation ». Quand un processus distant 212 se termine, la navigation ramène au processus appelant. Ce mécanisme de processus distants permet de partager sans difficultés un processus entre plusieurs utilisateurs, services ou fonctions. Par exemple, la prise de commande peut se faire par un commercial éloigné ou proche, aussi bien que par un service d'administration des ventes local.

Un sous processus standard 213 correspond à une procédure élémentaire totalement répétitive et commune à plusieurs processus. Ces processus standards 213 peuvent être définis pour une entreprise, compte tenu de ses besoins particuliers, pour un ensemble d'entreprises, auxquelles s'imposent les mêmes logiques ou règles de base. Par exemple, un calcul de stock réel ou prévisionnel peut être réalisé au magasin, aux achats, dans un calcul automatique de sortie de stock, lors d'un ajustement d'inventaire.

Un applicatif générique complexe 214 est commun à toutes les transactions réalisées par les utilisateurs. Il fait partie des fonctionnalités offertes par le procédé objet de l'invention :

- saisie simple, par exemple saisie ou mise à jour d'une fiche client, d'une fiche produit,
- saisie de liste, par exemple saisie ou mise à jour d'une fiche nomenclature (liste des composants attachés à une nomenclature),
- déclenchement d'alerte dans les processus de pilotage, par exemple dépassement de délai, écart anormal entre budget prévisionnel et réalisé,
- impression, par exemple, impression de certaines données clients : un client, une catégorie de clients, la totalité du fichier client (liste des clients avec leurs adresses, ou leurs chiffres d'affaires, leurs commandes en cours, ...).

Tâches complexes d'aiguillage 22. A chaque nœud de l'arborescence, les aiguillages entre plusieurs branchements sont déclenchés par trois voies : un choix 221 de l'utilisateur, par une navigation avec un menu, une identification d'une situation précise 222 comportant plusieurs possibilités de traitement (par exemple, dans le calcul de prix de revient d'un produit à partir de sa nomenclature, les calculs

du coût d'un produit acheté et celui d'un produit fabriqué sont différents et donnent lieu à des sous-processus différents), une évaluation d'une situation 223 sur la base de critères de contrôle avec déclenchement d'alertes, de boucles de feed-back (par exemple coûts ou délais anormaux).

Les tâches simples 23 de calculs ou de traitements d'information se caractérisent par le fait qu'elles seules traitent directement les informations. Le procédé de l'invention inclut les informations utiles à la conduite et à la maîtrise des flux et prévoit le pilotage et l'aide à la décision opérationnelle. Le dispositif utilise des capteurs et des règles de contrôle au plus près des événements. Le processus recueille les informations dès qu'un événement survient, en temps réel, et déclenche en tant que de besoin des signaux d'alerte auprès des personnes chargées des décisions opérationnelles, dans les délais utiles (définis par le diagramme) pour chacune d'elles (immédiat pour les décisions d'exécution, ou avec un délai approprié à la bonne efficacité des décisions de gestion).

Les calculs 231 comportent les opérations de base, avec prise en compte du nombre de décimales, les cumuls et ventilations et l'attribution de valeurs alphabétiques ou numériques.

Les comparaisons 232 sont effectuées entre des données alphabétiques ou numériques. D'après le résultat d'une comparaison, on déclenche des branches de traitement appropriées.

L'envoi de message 233 est effectué vers l'utilisateur ou vers d'autres utilisateurs (alerte), en fonction de résultats constatés à un poste.

Les transferts 234 sont relatifs à des liens entre des champs d'un écran ou d'un fichier avec des champs d'autres écrans ou fichiers, soit en importation, soit en exportation. Un transfert ponctuel peut aussi être effectué entre des champs d'origine et des champs de destination.

Le traitement de dates 235 concerne l'attribution de la date et de l'heure « système », le calcul de dates de début et de fin d'opérations d'après un délai ou calcul d'un délai d'après des dates de début et de fin, et la comparaison de dates.

Les tâches de communication et d'échange 24 avec les utilisateurs se font par l'intermédiaire d'un terminal informatique, soit à travers un réseau classique, soit par un navigateur internet. Le moteur 71 comporte un superviseur chargé de reconnaître la configuration matérielle et de communication (réseau local, intranet, internet, ...)

de l'utilisateur, et il est donc inutile de générer celle-ci dans l'applicatif. La gestion des échanges se fait par des tâches sur l'ensemble d'un écran, sur certains champs ou par des messages envoyés à l'utilisateur par le déroulement du processus.

Les tâches de communication et d'échange 24 comportent :

- la création et destruction d'un écran (activation de l'écran et affectation de mémoire) 241,
- l'affichage d'une page complète 242, avec son titre, ses libellés et ses champs d'information puis effacement de la page,
- l'affichage d'informations dans des champs 243 et modification d'écran, y compris affichage de l'ensemble des informations (par exemple à la lecture d'une fiche), l'affichage de champs modifiés (par exemple à la suite de calculs déclenchés par la saisie d'un champ et influençant d'autres champs, attribution de valeur dans un bouton radio, ...)
- la gestion des autorisations de modification 244, sur les constituant des codes ou noms servant à lire des fiches sur la base de données (accès en mode de sélection, champs de clés modifiables ou non), ou sur les informations courantes de la page d'écran (accès en mode de mise à jour, accès en simple consultation ou champ sans objet contextuel),
- le positionnement particulier sur un champ 245, y compris champ obligatoire à remplir et champs à corriger en cas d'erreur de saisie reconnue,
- l'envoi de message 246, y compris information à l'utilisateur, en particulier en cas d'erreur, l'affichage d'une boîte de dialogue à acquitter avant reprise de la saisie d'écran, ou l'affichage d'un message d'information durable en bas de l'écran,
- l'effacement de l'écran 247, y compris passage provisoire à un autre écran et, en fin d'utilisation d'un écran, avant sa destruction.

On observe que le dessin des écrans est adapté à chaque catégorie d'utilisateurs, en fonction des besoins de chacun. Leur simple description dans le référentiel applicatif suffit à faire fonctionner les tâches de gestion des échanges indiqués ci-dessus. Il est facile et rapide de les adapter en tant que de besoin, par simple modification de leur descriptif. Les informations sont présentées sous forme d'objets graphiques et spécifiées comme telles (champ de saisie ordinaire, boutons « base de données », boutons radio verticaux ou horizontaux, cases à cocher,

languettes, éditeur, table, code à barres, libellé, lien hypertexte). Les champs d'informations sont accessibles en mise à jour, accessibles seulement en consultation ou invisibles à destination par exemple de champs d'identificateurs, de champs intermédiaires de calculs, ou de champs d'informations élaborées à destination de tableaux de bord distants.

Concernant les droits d'accès, tout accès aux processus est soumis à un contrôle des droits de la personne en ligne. Chaque personne accédant à l'applicatif ne peut « voir », modifier, que ce qui lui est attribué par les droits qu'elle a reçu avec son code d'accès. Ces droits peuvent changer dans le temps, sous la responsabilité d'un gestionnaire des accès. Une personne ayant des droits élevés a intérêt à recevoir plusieurs codes d'accès avec des droits de plusieurs niveaux, afin de limiter les risques d'accès intempestifs, par d'autres personnes, à des informations confidentielles.

La présentation des documents imprimés est considérée comme celle d'un écran, avec la limite d'un accès à consultation, avec la possibilité de disposer de dimensions de « pages » plus grandes.

Les tâches de transaction avec des bases de données 25 comportent :

- l'initialisation et la libération des espaces d'informations liés à un fichier 251,
- la création ou la mise à jour de fiches 252 par écriture sur la base destinataire (chaque fiche reçoit automatiquement un numéro d'identification univoque qui sert de lien entre des items apparentés, sans que des modifications de code ou d'appellation aient une quelconque conséquence,
- la création de clés d'accès pour lecture de fiche particulière 253 : accès possible par des codes, des noms, des périodes de validité, des liens avec d'autres fiches, ... (l'accès à la base de données en multi-codification permet de rendre les informations accessibles à chaque catégorie d'utilisateurs en respectant sa culture propre, par exemple les codes produits sont souvent différents pour la technique et pour le commercial, voire entre plusieurs établissements; entre plusieurs processus autonomes (exemple : liaisons entre les différents services fournis par une banque à un même client).
- la lecture d'une fiche 254 (la fiche est lue sur une ou plusieurs clés d'accès à préciser : code, nom, identificateur, ...

- la lecture d'une liste de fiches ayant un rattachement commun 255 (par exemple les mouvements de stocks rattachés à un produit, composants d'un produit, lignes de commande, liste des commandes en cours pour un client, liste des commandes pour un produit),
- la suppression d'une fiche 256, selon des modalités de contrôle prédéterminées (par exemple, ne pas supprimer une fiche produit si elle possède un stock ou si une ligne de commande en cours y est rattachée),
- le parcours d'un fichier pour opérer un traitement systématique sur toutes les fiches, sur un ensemble de fiches entre deux bornes, ou sur des fiches ayant une même racine (par exemple, impression de tout ou partie d'un fichier, valorisation de marges sur toutes les commandes d'un même client, d'un agent commercial, d'une région, ...

Le moteur 71 gère la ou les bases de données d'après le descriptif des fichiers de données fourni par le référentiel applicatif, c'est-à-dire la liste des informations contenues dans chaque fiche et la liste des index d'accès, avec la liste des champs servant à constituer chacun de ces index, les liens entre des codifications multiples d'un même item dans plusieurs services (commercial et technique), plusieurs sites ou plusieurs entreprises (clients – fournisseurs, sociétés fusionnées, ...), les bases de données étant alors synchronisées selon une fréquence déterminée par le diagramme ou à la demande ou avant certains événements (par exemple pour réaliser un devis client).

On observe, concernant l'évolution du référentiel applicatif dans le temps, que, dans certaines limites de dimensions de chaque fiche, contrôlées par le moteur 71, il est possible de compléter les informations contenues dans un fichier. Si les formats des informations utilisées dans l'entreprise se modifient dans le temps, il est aisé de maintenir les données acquises dans le passé par une procédure système du moteur 71.

La bibliothèque de liaison 26 est partagée entre le référentiel applicatif et le moteur 71. Les informations nécessaires à l'exécution des tâches doivent être spécifiées en tant que de besoin et le moteur 71 les rattache à sept types :

a) menus et choix 261,

b) messages et alertes 262,

c) champs d'information et format de données 263,

d) désignations des informations 264,

e) fichiers 265,

f) interfaces visuelles 266, écrans ou pages internet,

g) branches de processus et tâches 267.

Les menus et choix 261. Les tâches d'aiguillage déclenchées par les utilisateurs comportent des informations de type « menus et choix ». Les utilisateurs se voient proposer un menu à chaque fois que leur intervention est nécessaire pour choisir entre plusieurs sous-processus et poursuivre la navigation. Le procédé nomme « choix » chacun des éléments du menu. A chaque choix est lié une branche du processus. Chacune des branches est dotée d'un code d'accès : si l'utilisateur ne possède pas les droits nécessaires, le menu filtre et ne propose par le choix correspondant..

Les messages et alertes 262. Le procédé nomme « messages » les échanges déclenchés au cours de transactions avec un utilisateur. Le procédé envoie, exclusivement à l'utilisateur, des messages d'information pour faciliter sa tâche et des messages d'anomalie et d'erreur lorsque celles-ci sont détectées. Le procédé nomme « alerte » l'envoi d'informations de pilotage à tous les utilisateurs concernés lorsqu'il détecte des anomalies, au cours d'un processus. Les alertes, envoyées à distance, sont d'une autre nature que les messages et sont rattachées à la procédure standard de gestion des alertes. Les tâches d'échanges et de communication avec les utilisateurs se font en temps réel.

Les champs d'information 263. Chaque élément d'une liste d'informations dans un fichier et dans une interface visuelle est nommé un « champ ». Pour les champs d'un fichier, la spécification du format de chaque information indique la taille de l'information et réserve en conséquence la place nécessaire pour les stocker sur disque. Lors de la génération du logiciel, la base de données est automatiquement créée ou mise à jour. Pour les champs d'un écran, d'une page internet ou d'un imprimé, leur présentation peut être faite suivant la demande de l'utilisateur et changée aisément.

Chaque champ possède un nom et un format. Chaque entreprise utilise un ensemble de données de base : références techniques et commerciales des produits, prix unitaires, montant d'une facture, temps, coûts, délais d'une commande, chiffre d'affaire hors taxes et toutes taxes comprises, ...

Le procédé nomme « format des données », la spécification du format de chacune de ces données. Ce format des données est utilisé pour réserver la place nécessaire dans les fichiers, pour afficher correctement les informations et pour contrôler la saisie des informations. L'attribution d'un format à une information spécifie sa présentation et sa dimension : type attribué (donnée alphanumérique, code numérique, nombre, date, ...), description (nombre de caractère, cadrage, présence éventuelle d'un signe, nombre de chiffres de la partie entière et nombre de décimales, numéro du mois inférieur ou égale à 12, numéro de la semaine inférieur ou égal à 54, jour dans le mois inférieur ou égal à 31, heure inférieure ou égale à 24, ...

Le procédé gère une codification multiple (multi-codification) pour tenir compte de références différentes pour un même objet, par exemple références techniques et commerciales, liens entre processus distincts, références différentes entre plusieurs sites ou sociétés appartenant au même groupe, ou plusieurs entreprises liées économiquement (clients – fournisseurs, sociétés fusionnées, ...).

Désignation des informations 264. Le procédé identifie par une désignation les informations présentées sur les supports de communication avec les utilisateurs. Chaque information d'une interface est présentée avec une désignation. Plusieurs désignations peuvent être utilisées pour tenir compte des différences de langue ou de fonction des utilisateurs.

Fichiers 265. Le procédé définit la structure des informations dans les fichiers et l'organisation de leur stockage en base de données. Les fichiers recueillent des ensembles d'informations structurées. Une fiche est une structure de fichier (par exemple fiche-client) comporte une liste d'informations définies par un nom et un format. Chaque fiche d'un fichier peut être atteinte par plusieurs clés d'accès différentes suivant la préoccupation de l'utilisateur, ou son besoin immédiat. Les événements sont identifiés en temps réel avec des étiquettes appropriées à la situation vécue. La base de données constitue une base de connaissances en permanence à jour.

Interfaces visuelles 266, écrans ou pages internet. Le procédé définit l'organisation des informations pour leur présentation aux utilisateurs. Les interfaces visuelles représentent un ensemble d'informations qui sont affichées avec leur désignation sur un écran d'ordinateur, une page internet, un imprimé. Comme pour

les fichiers, une interface visuelle comporte une liste d'informations définies par un nom et un format, ainsi qu'un lien avec les données correspondantes stockées dans les fichiers. Chaque information est dotée d'attributs précisant leur présentation visuelle, leur mode de traitement ou de saisie. Les interfaces visuelles peuvent être présentées sous une forme appropriée à chaque utilisateur, et modifiées pour répondre à de nouveaux utilisateurs ou de nouveaux besoins.

Branches de processus et tâches 267. Chaque branche de processus est représentée dans un diagramme et reflète la description des flux réels, obtenue avec le référentiel applicatif : nom du processus, enchaînement des opérations, nœuds de branchements, droits d'accès au processus (services, postes agréés). Chacune des opérations doit être décrite : nom de l'opération, nom de la tâche du moteur 71 qui devra être exécutée, paramètres de l'opération, suivant la syntaxe déterminée pour chacune des tâches existantes.

Le référentiel applicatif.

Ce référentiel est créé en association avec les utilisateurs concernés par le processus à informatiser, d'après la situation existante, et en étudiant d'éventuelles améliorations, facilitées par l'emploi d'un outil informatique. Une fois l'accord des utilisateurs obtenu, le référentiel est numérisé (par exemple, par saisie ou par lecture optique ou sonore), en vue d'obtenir une importation automatique dans le logiciel exécutable.

Générateur de logiciel exécutable sur mesure 20. Ce générateur utilise les données du référentiel applicatif, transcrites sous forme numérique. Le générateur 20 procède en deux étapes :

- importation 361 (voir figure 3) des données du référentiel dans une base de données interne, et
- génération 362 (voir figure 3) du logiciel exécutable.

On observe, en figure 3, que la mise en oeuvre d'un mode particulier de réalisation du procédé objet de la présente invention comporte le mise en oeuvre d'un logiciel système commun à tous les logiciels applicatifs et :

- une étape d'initialisation 31 d'un système informatique,
- une étape de représentation 32 du processus, mettant en oeuvre un très petit nombre de classes d'actions ou objets génériques, typiquement inférieur à vingt, dans au moins un diagramme de l'application,

- une étape de transcription 33 de chaque objet de chaque diagramme en une action correspondant à un objet doté d'attributs, chaque classe d'action ou objet générique étant associé à une interface de saisie de données d'application, comportant, pour programmer chaque action :

i) une étape de nommage 331 au cours de laquelle on donne un nom à ladite action,

ii) une étape de définition de fonction 332, au cours de laquelle on met ladite action en correspondance avec une tâche, et,

iii) une étape de définition d'information 333, au cours de laquelle on désigne les informations qui seront traitées dans l'action.

- une étape de transcription 34 des noeuds, embranchements et feuilles de chaque diagramme en une action correspondant à un objet doté d'attributs,

- une étape de pré-compilation 35 au cours de laquelle on vérifie que les attributs des objets nécessaires pour la logique de fonctionnement de l'application existent et sont convenablement fournis, en syntaxe,

- une étape de compilation 36 au cours de laquelle les descriptifs de données des objets dotés d'attributs sont intégrés et sont assemblés avec le logiciel système, pour obtenir un logiciel applicatif exécutable, et

- une étape d'exécution 37 du logiciel applicatif exécutable compilé au cours de l'étape de compilation 36.

Au cours de l'étape de transcription d'objets, une ou plusieurs action(s) lance(nt) un traitement complet se trouvant à un endroit distant d'une arborescence correspondant audit au moins un diagramme, et une fois celui-ci terminé, retourne à son point de départ.

Préférentiellement, au cours de l'étape d'exécution, le logiciel applicatif exécutable met en oeuvre une bibliothèque de gestion du déroulement des processus correspondant audit au moins un diagramme, ladite bibliothèque constituant un automate 70 qui gère le déroulement des processus et exécute les opérations qui les jalonnent, le déroulement des opérations étant défini, dans le référentiel applicatif, à l'aide de la méthode 10, en décrivant les flux réels.

Préférentiellement, au cours de l'étape de compilation ou au cours de l'étape d'exécution, il met en oeuvre le moteur 71 qui comporte un superviseur chargé de reconnaître la configuration matérielle et de communication. Préférentiellement, le

moteur 71 gère une ou plusieurs bases de données d'après un descriptif des fichiers de données fourni par le référentiel applicatif, c'est-à-dire une liste des informations contenues dans chaque fiche et la liste des index d'accès, avec une liste des champs servant à constituer chacun de ces index, les liens entre des codifications multiples d'un même item dans plusieurs services, plusieurs sites ou plusieurs entreprises. Les bases de données sont synchronisées selon une fréquence déterminée par le diagramme, à la demande ou avant certains événements prédéterminés.

Préférentiellement, l'étape de compilation (36) remplace le nom d'action donné par le transcripteur, au cours de l'étape 33, par un index dans une table de tâches.

Préférentiellement, au cours des étapes de représentation et de transcription, ledit au moins un diagramme correspond à au moins une arborescence dans laquelle les nœuds et feuilles, où le code est effectué, sont composés d'actions, les valeurs de retour de ces actions déterminant le déplacement dans l'arborescence.

Une fois terminée la génération, le procédé peut passer sur l'applicatif de l'entreprise, chaque processus de l'entreprise étant accessible suivant les droits des utilisateurs.

Le référentiel applicatif peut être modifié. Une nouvelle génération du logiciel exécutable met à la disposition des utilisateurs la nouvelle version. Le temps nécessaire à une génération est de l'ordre de dix à trente minutes, suivant l'importance du processus.

Etape d'importation 361 des données du référentiel dans une base de données interne. Au cours de cette phase, le générateur 20 contrôle la synthaxe et la cohérence des données fournies par le descriptif de l'application. La présence de certaines données de description obligatoires est contrôlée. Les données appelées dans les traitements doivent exister : champs d'informations, libellés, messages, processus, tâches. Si le générateur 20 détecte une erreur, il envoie un message et refuse de passer à la phase suivante.

Etape de génération 362. Une fois le référentiel applicatif accepté, le générateur 20 génère le logiciel applicatif en associant le moteur 71 et les données applicatives. La fin de cette phase de génération 362 redonne la main à l'utilisateur, avec la nouvelle version disponible en cas de changements.

On définit deux classes de structures :

- celles qui servent à décrire les informations utilisées dans les applications, particulièrement le format des données, la structure des enregistrements dans la base de données et la façon dont les écrans sont présentés, et
- celles qui correspondent à la façon dont le programme va se dérouler (l'ordre dans lequel les traitements vont être effectués) : diagramme de flux ou de processus et actions sur les flux.

Les premières sont des données de description et les secondes des données de déroulement.

Les données de déroulement comportent :

a/ les actions

Les données de déroulement ont une organisation particulière qui a une forte incidence sur la façon de programmer les applications. Une des motivations est d'éviter la programmation directe en langage informatique, par exemple en langage C. A cet effet, un certain nombre de fonctions de manipulation et de présentation des données ont été définies. Leur ont été associées un certain nombre d'arguments, fixes ou pouvant être paramétrés grâce à certaines structures de données (les « cartes » décrites plus loin) et d'empaqueter le tout dans un ensemble identifié par un code. Chaque ensemble porte le nom d' « action ». La structure action est donnée ci-dessous :

```
struct action
{
        char            *ac_name;
        int             *ac_fonc
        unsigned long ac_param[MPRM];
        char            *ac_article;
}
```

Programmer une action, dans cet esprit, revient simplement à lui donner un nom, affecter la "tâche moteur" ou "tâche type" (par exemple : calcul, comparaison, contrôle, branchements, ...) et désigner les informations qui seront traitées dans l'action.

Les actions sont enchaînées suivant leur ordre fonctionnel dans des diagrammes. Les actions ramenant toutes à des codes définis, il est possible de représenter ainsi les structures de contrôle classiques telles que branchement,

boucle, test, etc ... Ces contrôles sont eux-mêmes effectués par des actions, voir plus loin. Déroulement du code dans l'application – Diagrammes de processus.

Le concept sur lequel le déroulement est basé, intimement lié à celui d'action, est celui d'arborescence. Ce n'est pas une idée nouvelle qu'une application à base de menus soit structurée en forme d'arborescence. Dans notre cas cependant, le diagramme a ceci de particulier que ses nœuds et feuilles, où le code est effectué, sont composés d'actions. En fait, chaque nœud est un réceptacle où peuvent tenir MACT actions – Le nombre d'actions dans une branche, MACT, est un paramètre du moteur 71. Les valeurs de retour fournies par ces actions, déterminent le déplacement dans les arborescences.

On observe qu'assembler un certain nombre d'actions standards en des diagrammes donne un résultat plus simple à contrôler que du code C généré directement. Le principe étant que des actions, supposées travailler sur des données convenablement encapsulées, sont considérées comme sûres, et leur contenu n'a jamais à être tracé.

Implémentation des diagrammes.

Les noms des diagrammes et les actions jalonnant les diagrammes sont placés dans une structure appelée struct branche. La structure branche est décrite ci-dessous :

```
typedef struct branche
{
        char            nom_digramme[LNOM];
        unsigned int    table_traitements[MACT];
        char            *securité ;
} node;
```

Le nom "nom_diagramme" permet d'identifier le diagramme et de se déplacer de diagramme en diagramme. Les éléments de table_traitements sont des actions qui seront exécutées à l'arrivée sur le diagramme; le concepteur leur donne un nom et la compilation remplace ces noms par des index des actions dans une table qui les regroupe : leur accès est instantané et permet d'obtenir de bonnes performances des traitements informatiques.

La chaîne de caractères "securité" est examinée avant toute demande d'arrivée sur le diagramme, pour permettre d'interdire l'accès en fonction des droits

de l'utilisateur de l'application. C'est ainsi que certains menus ne seront pas visibles pour certains utilisateurs, et visibles pour d'autres.

On décrit ci-dessous, sur un exemple, la façon dont on se déplace dans les nœuds et dont on exécute le code.

Supposons que le nom du diagramme courant soit « L ».

En arrivant sur ce diagramme, le système exécute l'action tab_tr[0]. Si elle ramène le code NEXT, comme le font, par exemple les 4 premières actions du diagramme L11, on exécute l'action suivante. Si elle ramène DSCD (pour « descend »), elle doit aussi avoir positionné une variable globale, Choix. Cette variable va alors être concaténée au nom du diagramme courant, pour donner le nom du nouveau diagramme. Par exemple, si la variable choix vaut « 1 », on se dirige vers le diagramme « L1 ». On note DSCD(1) l'association de DSCD et de choix=1. L'inverse de la procédure se produit si une action retourne RMTE. Dans ce cas, l'action doit avoir positionné certains drapeaux (« flags ») internes pour déterminer à quelle action de la branche de remontée on reprend le traitement. Par exemple, une remontée à l'action 3 depuis L1 nous positionne sur la quatrième action du nœud L (les actions étant numérotées à partir de 0.)

Plus précisément, c'est le caractère ASCII représentant le nombre choisi qui est additionné. Il faut donc éviter de donner à "Choix" une valeur en dehors de l'intervalle [a-zA-Z-0-9], afin d'éviter les caractères invisibles ou dotés de fonctionnalités dans la table ASCII.

On peut enfin passer à une action du même nœud sans que ce soit nécessairement la suivante en positionnant NACT plus le numéro d'ordre de l'action à atteindre par la branche Choix et en retournant NACT. Le traitement se poursuit tant que l'on n'est pas passé sur une action de sortie, dont la fonction associée est de fermer les fichiers restés ouverts dans la base de données et de libérer les espaces mémoires.

On observe que cela ne signifie pas forcément que l'on est revenu à la racine ou que l'on a parcouru tout le diagramme ou toute autre condition de cet ordre. Il est aussi possible de sortir « en urgence » de l'application avec certaines routines de traitement d'erreurs qui envoient un message et ferment ou clôturent la session.

On comprend que le système, lors du passage dans les menus, interprète les entrées clavier. Si une touche de fonction (de F1 à F9 dans la version standard) est

tapée, on descend immédiatement sur la branche courante; à laquelle on concatène un caractère entre « 1 » et « 9 ». n peut ainsi tracer le chemin parcouru depuis le début. Il existe par ailleurs une variable globale de débuggage, qui affiche, lors des parcours des diagrammes, le nom de la branche courante en bas d'écran et permet de contrôler le bon déroulement de la procédure et d'identifier rapidement les erreurs commises dans la numérisation des processus.

Cependant, si une action positionne, par exemple, Choix à « 7 », rien ne permet de savoir, à la lecture des sources du diagramme, si la branche obtenue est atteinte suite à une entrée clavier, ou suite à un traitement. On conseille au concepteur de réserver des valeurs de Choix menu dans l'intervalle [1-9] aux branchements obtenus dans un menu, et d'affecter des valeurs alphabétique pour les branchements déclenchés automatiquement par le contexe applicatif.

On note que si la dernière action d'un diagramme ramène la valeur NEXT, le système trouve une erreur et s'arrête, de même que si on cherche à descendre lorsque l'on se trouve déjà sur une feuille, ou si on cherche à remonter en étant déjà à la racine.

Lancement des diagrammes.

Le nom du premier diagramme lancé dépend de l'activité désirée. Il est lancé à la fin du main par un appel à la fonction d'interprétation de diagramme. Cette routine prend bien sûr le nom du diagramme en argument, mais aussi le numéro de l'action qui est exécutée en premier. En effet, on veut parfois éviter de débuter à l'action 0, les premières actions pouvant par exemple effectuer des initialisations indésirables. On donne aussi les arguments type_arbre et no_arbre qui donneront leurs valeurs aux variables tp_arb et no_arb. Ces variables permettent de déterminer le type de diagramme, c'est-à-dire s'il s'agit d'un diagramme de Saisie, de Liste ou d'Impression, pour parler des types standards, ou un autre type si besoin est. Comme plusieurs modules peuvent avoir le même type (par exemple, les modules clients, fournisseurs et articles peuvent être tous des Saisies), on les distingue par un numéro d'ordre no_arb. Ce numéro dépend de la génération des cartes par le générateur 3. Ces variables permettent de relier les tables de paramètres personnalisés d'un module à un processus commun standard (voir saisies de fiches simples ou de fiches de listes, impression, déclenchements d'alerte).

Lancement de diagrammes particuliers

Il est parfois intéressant, depuis une action donnée, de lancer un traitement complet se trouvant à un endroit distant de l'arborescence, et une fois celui-ci fini, de se retrouver à son point de départ. Cela est rendu possible par le fait que, bien que tous les diagrammes de l'application soient regroupés dans un même tableau, celui-ci regroupe en fait une forêt. On peut lancer une arborescence nouvelle comme un lien avec un « sous-processus » ou "sous-système" en terme cybernétique, à l'aide de l'action de lancement d'un diagramme, tout comme « main() » lance le diagramme initial. C'est même ainsi que les divers modules sont intégrés à l'application. On utilise un diagramme existant décrivant un module, on lui donne un nom, et, depuis notre arborescence d'origine, on lance le diagramme en question. On sort de ce diagramme particulier par une action ramenant ENDA.

Diagrammes clients et diagrammes communs

Les diagrammes clients sont des diagrammes spécifiques à une application. Les diagrammes communs sont des « objets procéduraux », permanents et accessibles par l'appel à des cartes identifiant les travaux à réaliser et les informations nécessaires pour obtenir le déroulement d'une procédure spécifique suivant un modèle standard (objet procédural) : nom des fichiers et écrans, noms des menus et messages, diagrammes de traitement à exécuter à des moments précis de la procédure (par exemple : fin de la sélection d'une fiche à mettre à jour). Certains ensembles de traitements vont être utilisés dans toutes les applications. Par exemple, les processus de saisie de liste, de Saisies, d'impressions, etc ... sont communs. Dans le cas des Saisies, par exemple, on a toujours un menu proposant un choix de type :

CREATION
MISE A JOUR
CONSULTATION
SUPPRESSION
FIN

puis, après sélection de MISE A JOUR, on a un choix, par exemple :

PREMIER
DERNIER
SELECTION
FIN

puis, après sélection de PREMIER, par exemple :

PREMIER
DERNIER
SUIVANT
PRECEDENT
SELECTION
VALIDATION CHOIX
FIN.

Les déroulements de ces diagrammes sont fixés de façon permanente dans des procédures standard. Il suffit de fournir les identificateurs d'écrans et de fichiers utilisés, et ces diagrammes vont les visualiser, modifier les menus au fur et à mesure des étapes, etc ... Le mécanisme de la procédure est toujours identique, mais son contenu est adapté, sur mesure, aux besoins exprimés par les utilisateurs. Bien entendu, un certain nombre de points d'entrée subsiste pour insérer des actions particulières à chaque client, par exemple si le mode de lecture des données est très particulier (par exemple, si le fichier n'est pas un fichier standard mais est importé d'un autre système, la façon de lire la première fiche n'est pas standard, donc doit être donnée par le concepteur). Par contre, le fait d'avoir le choix de demander la première fiche reste standard, donc le diagramme considéré est utilisable. L'ensemble des informations propres à chacun des applicatifs est réuni de façon structurée dans des "cartes" (voir ci-après) : le concepteur "câble" une carte pour approprier le processus standard à chaque processus particulier

Les traitements particuliers à certains clients qui ne sont pas généralisables sont codés sous forme de diagrammes spécifiques et d'actions. Ceux-ci ont exactement la même structure que les diagrammes standards, mais sont placés dans un tableau secondaire, donc ne sont pas compilés avec les applications n'y accédant pas. Notons que la recherche des noms diagrammes par le système commence toujours par les diagrammes clients. Il est ainsi possible de définir son propre diagramme client et de lui donner le même nom qu'un diagramme commun dont on n'apprécie pas le comportement. C'est le diagramme client qui sera pris à la place. On ne peut, en revanche, demander explicitement à utiliser la version client ou la version commune d'un diagramme.

Lancement d'actions supplémentaires

Parfois, le nombre d'actions maximal d'un diagramme n'est pas suffisant pour effectuer un traitement. Il existe plusieurs méthodes pour régler ce type de problèmes. On peut aussi lancer une action particulière qui permet de se rebrancher sur un diagramme avec un retour à l'action suivant l'action de branchement (hypernavigation).

Sur chaque champ d'écran, on peut déclencher des diagrammes, à l'entrée du champ et à la sortie (attribution de valeurs, calculs instantanés, ...)

Cartes

Les cartes sont des ensembles de paramètres permettant d'orchestrer le comportement des diagrammes qui lui sont relatifs. Elles regroupent la plupart des informations utiles pour la gestion des divers modules constitutifs d'une application, par exemple le nom des fichiers concernés, des écrans de Saisie, d'en-tête et de liste si besoin est, les noms des champs de clé pour les lectures/écritures de données, des drapeaux (flags) de contrôle de traitements, les noms des menus à afficher en bas des écrans, les actions spécifiques à lancer dans le déroulement des diagrammes, etc ...

Une carte est une structure regroupant un certain nombre d'entrées numériques non signées, et un certain nombre de chaînes de caractères. Nous ne considérons que le cas des données numériques, la discussion étant similaire pour les cartes concernant des données de type « caractère ».

On distingue, en général trois types de cartes, correspondant aux trois grands types de traitement, les cartes de Saisies, les cartes de Listes et les cartes d'impression. La différence entre chaque type réside seulement dans le nombre et la signification des entrées de la structure le représentant.

L'identification de ces cartes est donnée par deux paramètres tp_arb et no_arb, connus à tout instant par l'environnement de navigation, et permettant de savoir dans quel type de module de l'application l'utilisateur travaille (saisie simple, saisie de liste, impression, ... et sur quel sujet : article, stocks ...).

Toutes les structures cartes d'un type donné sont regroupées dans un tableau (on a donc trois tableaux principaux), et une table de regroupement, cart[], regroupe les structures pointeurs de chaque tableau. L'ordre des divers types de cartes dans ce tableau est le suivant :

0 Réservé

1 Saisie des listes
2 Saisie simple
3 Réservé
4 Impressions
5 Réservé

On décrit ci-après comment référencer un paramètre dans une carte. Chaque donnée d'une carte est identifiée dans les actions standards sous la forme : type de paramètre (numérique ou alpha) x 1 000 + numéro d'ordre du paramètre dans le descriptif de la carte, par exemple 58. Le coeur du moteur 71 reconnaît que la donnée provient de la carte en cours et fait une double indirection vers la carte du processus en cours, puis vers la 58^{e} position dans cette carte. Ce traitement est automatiquement fait par l'exécuteur d'actions, fournissant ainsi à celles-ci les bonnes valeurs. Ce type d'interprétation est fait non pas par l'action elle-même, mais par le mécanisme d'interprétation des diagrammes du coeur du moteur 71.

Ce principe permet, par exemple, à une même action de visualisation d'afficher des écrans différents suivant la carte dans laquelle on se trouve, tout en conservant les mêmes paramètres d'appel. Il suffit d'avoir modifié auparavant la table de paramètres indexés.

Ce type d'interprétation est fait non pas par l'action elle-même, mais par le mécanisme d'interprétation des diagrammes ("câblage" de la carte). En d'autres termes, une action comporte des données soit décrites de façon spécifique soit décrites par des paramètres (carte + numéro dans la carte).

Si les champs sont des champs de clés, le nœud peut procéder à des tests pour vérifier leur validité (la fiche doit exister en mode "mise à jour" et ne doit pas exister en mode "création"). Les fonctions de saisie standard vont ensuite passer sur tous les champs sauf sur les champs de clés. Après la création de la clé ou la sélection d'une fiche en mise à jour, la procédure passe dans une phase de saisie courante et traite les champs accessibles.

Niveau

Une liste de structures chaînées allouées dynamiquement par le système permet d'obtenir, à tout moment, des informations intéressantes concernant l'état courant de l'application. Ces structures dites structures niveau, regroupent notamment :

- le nom du diagramme courant (nom)
- l'index de l'action dans le diagramme courant (nact)
- le nom de la carte (tp_arb)
- l'index de la carte dans la table des cartes (no_arb)

Le système alloue une telle structure à chaque descente ou à chaque lancement d'un diagramme. Les structures sont libérées en fin de diagramme ou après remontée.

Transferts, insertions et extractions

L'essentiel des traitements effectués consiste en transferts d'information d'une zone vers une autre, par exemple d'un champ de fichier, après la lecture d'un enregistrement, vers un champ d'écran permettant la visualisation de l'information considérée. Les structures d'écrans/fichiers permettent jusqu'à trois codages de transferts. Un codage tient sur un unsigned int (32 bits) et à l'aspect suivant :

type de la donnée (fichier, écran), nom du fichier ou de l'écran, nom du champ.

REVENDICATIONS

1 - Procédé de génération de logiciel applicatif de gestion d'un processus, caractérisé en ce qu'il met en oeuvre un logiciel système commun à tous les logiciels applicatifs et en ce qu'il comporte :

- une étape de représentation du processus (32), mettant en oeuvre un très petit nombre de classes d'actions ou objets génériques, typiquement inférieur à vingt, dans au moins un diagramme de l'application,
- une étape de transcription (33) de chaque objet de chaque diagramme en une action correspondant à un objet doté d'attributs, chaque classe d'action ou objet générique étant, pendant l'étape de transcription, associé à une interface de saisie de données d'application,
- une étape de transcription des noeuds, embranchements et feuilles (34) de chaque diagramme en une action correspondant à un objet doté d'attributs,
- une étape de pré-compilation (35) au cours de laquelle on vérifie que les attributs des objets nécessaires pour la logique de fonctionnement de l'application existent et sont convenablement fournis, en syntaxe,
- une étape de compilation (36) au cours de laquelle les descriptifs de données des objets dotés d'attributs sont intégrés et sont assemblés avec le logiciel système, pour obtenir un logiciel applicatif exécutable, et
- une étape d'exécution (37) du logiciel applicatif exécutable.

2 - Procédé selon la revendication 1, caractérisé en ce que, au cours de l'étape de transcription d'objets (33), au moins une action lance un traitement complet se trouvant à un endroit distant d'une arborescence correspondant audit au moins un diagramme, et une fois celui-ci terminé, retourne à son point de départ.

3 - Procédé selon l'une quelconque des revendications 1 ou 2, caractérisé en ce que, au cours de l'étape d'exécution (37), le logiciel applicatif exécutable met en oeuvre une bibliothèque de gestion du déroulement des processus correspondant audit au moins un diagramme, ladite bibliothèque constituant un automate (70) qui gère le déroulement des processus et exécute les opérations qui les jalonnent, le déroulement des opérations étant défini, dans le référentiel applicatif, à l'aide de la méthode (10), en décrivant les flux réels.

4 - Procédé selon l'une quelconque des revendications 1 à 3, caractérisé en ce que, au cours de l'étape de compilation (36) ou au cours de l'étape d'exécution (37), il met en oeuvre un moteur (71) qui comporte un superviseur chargé de reconnaître la configuration matérielle et de communication.

5 - Procédé selon la revendication 4, caractérisé en ce que le moteur (71) gère une ou plusieurs bases de données d'après un descriptif des fichiers de données fourni par le référentiel applicatif, c'est-à-dire une liste des informations contenues dans chaque fiche et la liste des index d'accès, avec une liste des champs servant à constituer chacun de ces index, les liens entre des codifications multiples d'un même item dans plusieurs services, plusieurs sites ou plusieurs entreprises.

6 - Procédé selon la revendication 5, caractérisé en ce que les bases de données sont synchronisées selon une fréquence déterminée par le diagramme, à la demande ou avant certains événements prédéterminés.

7 - Procédé selon l'une quelconque des revendications 1 à 6, caractérisé en ce que l'étape de transcription d'objets (33) comporte, pour programmer chaque action :

- une étape de nommage (331) au cours de laquelle on donne un nom à ladite action,
- une étape de définition de fonction (332), au cours de laquelle on met ladite action en correspondance avec une tâche, et,
- une étape de définition d'information (333), au cours de laquelle on désigne les informations qui seront traitées dans l'action.

8 - Procédé selon la revendication 7, caractérisé en ce que l'étape de compilation (36) remplace le nom d'action donné, au cours de l'étape de nommage (331), par le transcripteur, par un index dans une table de tâches.

9 - Procédé selon l'une quelconque des revendications 1 à 8, caractérisé en ce que, au cours des étapes de représentation (31) et de transcription (32, 33), ledit au moins un diagramme correspond à au moins une arborescence dans laquelle les nœuds et feuilles, où le code est effectué, sont composés d'actions, les valeurs de retour de ces actions déterminant le déplacement dans l'arborescence.

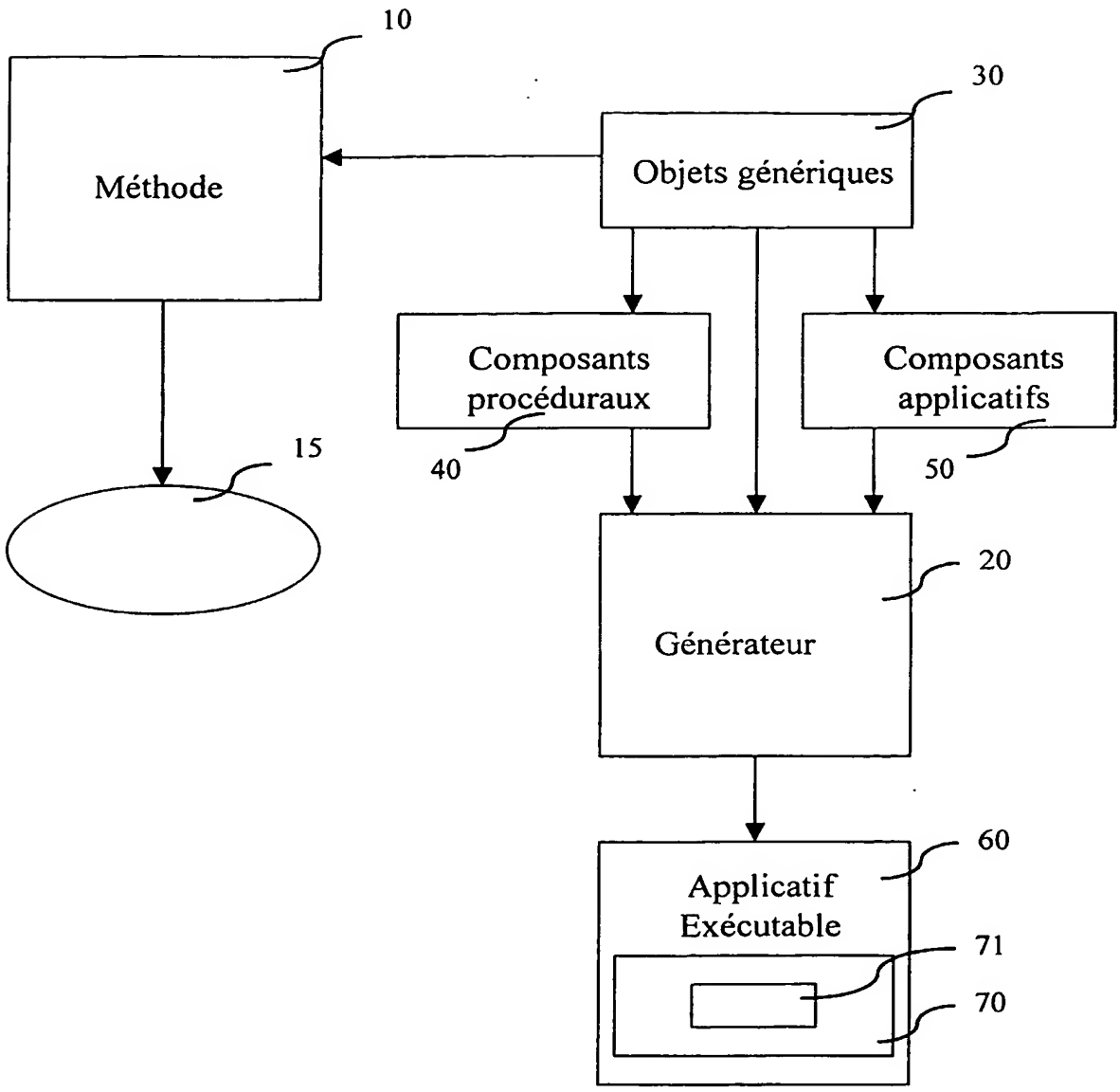

10
Méthode
30
Objets génériques
Composants procéduraux
40
Composants applicatifs
50
15
20
Générateur
60
Applicatif Exécutable
71
70

Figure 1

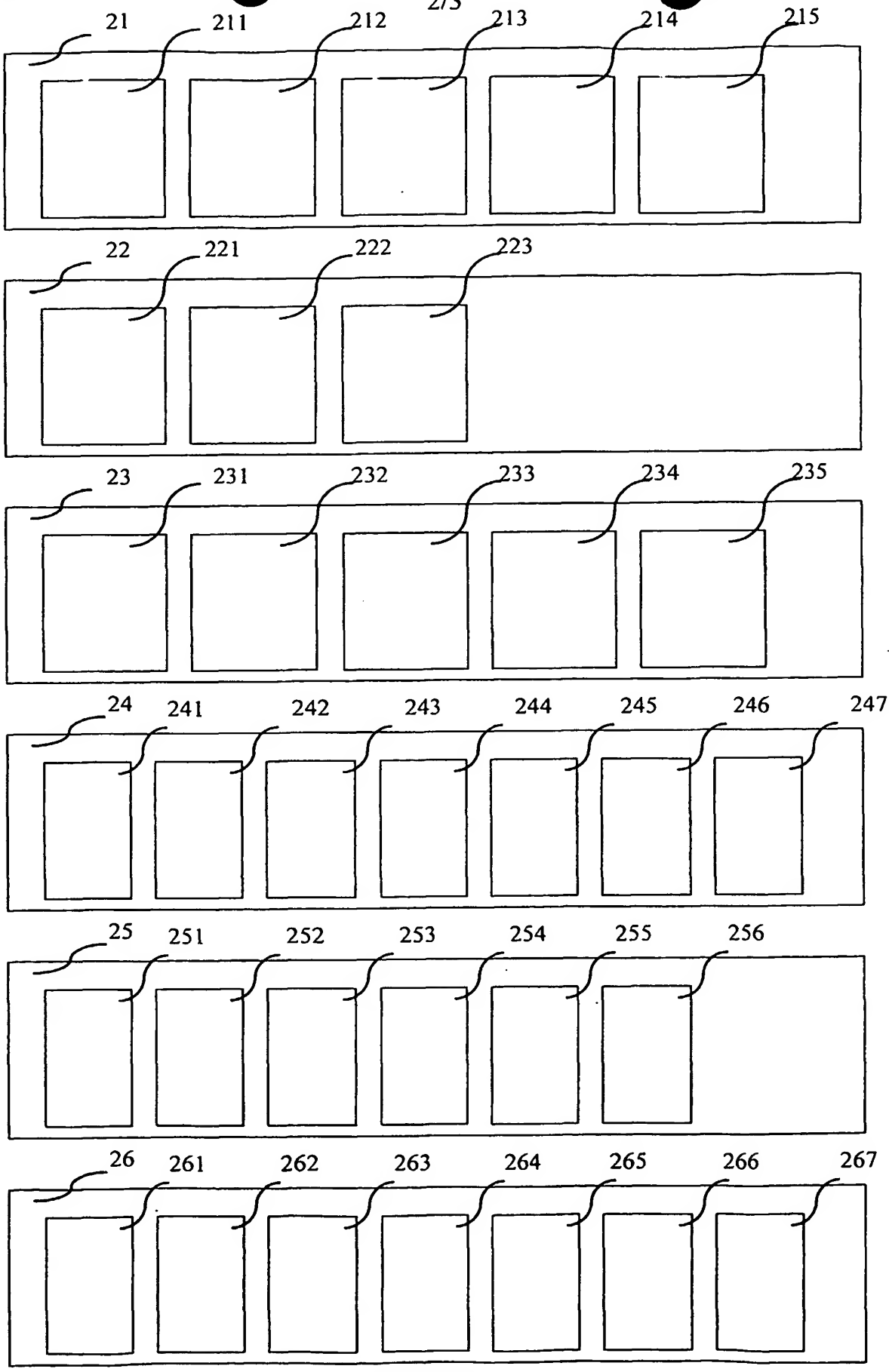

21
211
212
213
214
215
22
221
222
223
23
231
232
233
234
235
24
241
242
243
244
245
246
247
25
251
252
253
254
255
256
26
261
262
263
264
265
266
267

Figure 2

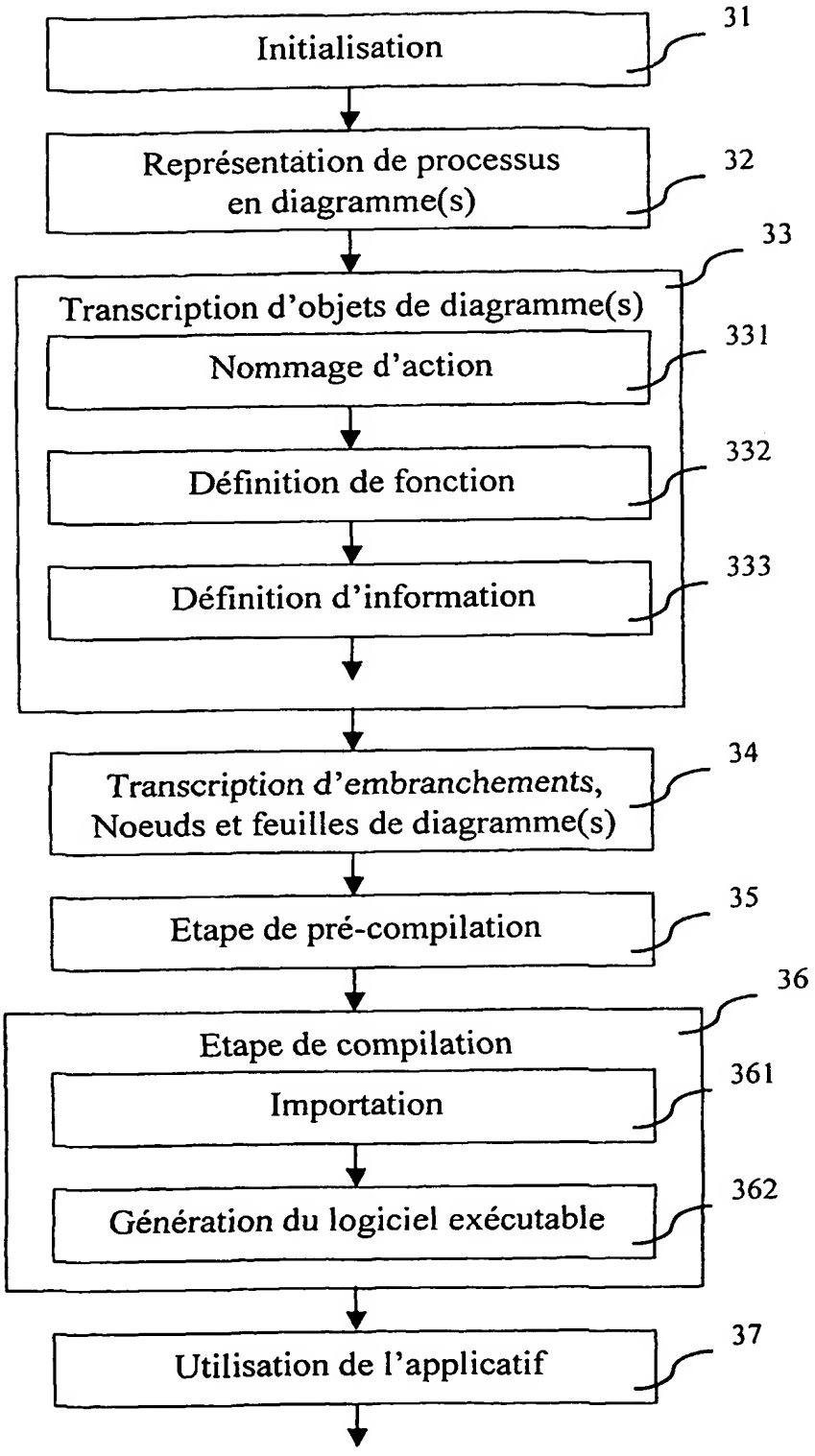

Initialisation
31
Représentation de processus en diagramme(s)
32
33
Transcription d'objets de diagramme(s)
Nommage d'action
331
Définition de fonction
332
Définition d'information
333
Transcription d'embranchements, Noeuds et feuilles de diagramme(s)
34
Etape de pré-compilation
35
36
Etape de compilation
Importation
361
Génération du logiciel exécutable
362
Utilisation de l'applicatif
37

Figure 3